Sommaire

Répète après moi ?

Il a de grandes oreilles mais est-ce qu'il nous entend ?

Un autre moyen de « commander » notre lapin est tout simplement notre parole, si on lui disait (gentiment) ce que nous voulons qu'il fasse, le ferait-il ?

Voilà un nouveau challenge intéressant, j'ai l'impression que vous êtes impatient.

Retour à notre site favori
http://dev.karotz.com/sdk/
Et lecture (compréhension ?) de l'instruction qui lui permet d'entendre (avant de comprendre)
Mais au fait, on cherche quoi exactement, quel est le nouvel acronyme que nous devons trouver ?
Il s'agit de l'ASR (Automatique Speech Recognition) quand je vous dis qu'il est grand temps de vous mettre à l'anglais ! En français reconnaissance automatique de la parole.

karotz.asr

start

```
karotz.asr.string(string grammar, string lang,  function(asrResult));
```

grammar : the asr grammar (root part)
lang : "fr-FR", "en-GB", "en-US", "es-ES", "de-DE"
function : callback

```
The grammar use ABNF format.
ABNF grammar W3C standard

#ABNF 1.0 UTF-8;
language fr-FR;
mode voice;
tag-format <semantics/1.0>;
root $App;
public $App = [lesson] 1 { $.param='1'} |  [lesson] 2 { $.param='2'}
|  [lesson] 3 { $.param='3'};
```

In this example : the text "lesson" is optional. The user can say "lesson 1" or "1". The semantic interpretation will return the semantic with param=1. Semantic is useful to build useful asr return.

Your grammar argument is encapsulated in a build grammar with headers (lang, …)

```
#ABNF 1.0 UTF-8;
language yourLang;
mode voice;
tag-format <semantics/1.0>;
root $App;
public $App = yourGram;
```

The callback function argument :

asrResult.confident : 0-100 (it's not a linear variable. more than 30 can be considered as a acceptable result.
asrResult.text : recognized text
asrResult.semantic : contain the semantic interpretation

in our example : asrResult.semantic.param = 1 or 2 or 3

Vous avez pris de l'assurance grâce à tous vos exercices aussi cette fois-ci je sais bien que vous n'êtes pas parti, j'en ai même entendu certains dire « même pas peur », peut-être celles et ceux qui sont passés programmeur 1er échelon ;)
Même pas peur alors on ne renouvelle pas les erreurs précédentes et on va de suite mettre à jour notre fichier descriptor.xml afin d'éviter de mauvaises surprises.
Et on en profite pour en mettre un maximum comme cela ce sera fait pour la suite des tutoriels, on va ajouter asr (reconnaissance vocale) mais également http (lire un site Internet, certainement dans un futur tutoriel), multimedia (écouter de la musique) et ears (bouger les oreilles).
Le voici, faites un copier/coller

```
<descriptor>
  <version>1.0.0</version>
  <accesses>
    <access>tts</access>
    <access>button</access>
    <access>rfid</access>
          <access>http</access>
          <access>ears</access>
          <access>ars</access>
  </accesses>
        <interruptible>true</interruptible>
        <awake>true</awake>
</descriptor>
```

Petite explication sur les 2 lignes supplémentaires ajoutées
<Interruptible> true signifie qu'on autorise une autre application à interrompre la notre (votre lapin a certainement une vie et laissons lui la vivre) mais si on souhaite être exclusif (le temps que notre application fonctionne) il

faudrait écrire *<Interruptible>false*
<awake> true signifie que si jamais le lapin s'était endormi pendant que nous lisions / comprenions ce tutoriel il serait bien qu'il se réveille lors de notre prochain test. Ces 2 dernières informations sont pour le futur car non traitées actuellement.
Ça c'est fait on va pouvoir passer aux choses sérieuses

Nous allons nous fixer un objectif avec ce tutoriel, réaliser un petit jeu, Karotz choisira un chiffre entre 0 et 9 et nous lui donnerons un chiffre, Karotz nous dira si nous sommes au dessus, en dessous dans ces 2 cas on refera une proposition et si c'est le bon chiffre nous aurons les félicitations du jury euh du lapin je voulais dire.

Procédons par étape, commençons par lui dire un mot et demandons lui de nous le redire.

L'instruction est :

```
karotz.asr.string(string grammar, string lang,  function(asrResult));
```

Si vous voulez vraiment vous faire peur, vous pouvez suivre le lien qui est donné concernant les explications de l'argument « grammar »
Sinon restons fidèles à nos habitudes et essayons de faire simple.
En fait ce qu'il faut comprendre c'est lorsqu'on veut parler à notre lapin nous devons utiliser des mots qui sont dans une liste à sa disposition, ça n'est pas un magnétophone que nous utilisons, aussi il a besoin d'analyser ce qu'il entend parmi une liste prédéfinie et bien entendu cette liste répond à des règles (pour Karotz le format ABNF)
Nous retiendrons l'essentiel, les mots attendu seront dans une chaîne de caractères donc la liste complète sera entre guillemets et chaque mot sera séparé par le caractère | (que l'on obtient en utilisant altgr + 6)
Je veux que Karotz reconnaisse les chiffres de 0 à 9 ma liste sera donc
"0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 "
On pourrait aussi écrire
"zéro |un|deux|trois|quatre|cinq|six|sept|huit|neuf"
Ça c'est pour string grammar
Pour string lang on utilisera "fr-FR" quoique pourquoi être franco_français, Karotz est Internationnal non mais voici une petite liste (non exhaustive bien entendu) mais attention malgré tout l'ASR Karotz n'utilise pas toutes les langues.

Vous êtes plutôt	Vous devez mettre
français de France	"fr-FR"
français du Canada	"fr-CA"
français de Belgique	"fr-BE"
français de Suisse	"fr-CH"
anglais du Royaume-Uni	"en-GB"
anglais des USA	"en-US"
anglais du Canada	"en-CA"
danois du Danemark	"da-DK"
italien d'Italie	"it-IT"
italien de Suisse	"it-CH"
espagnol d'Espagne	"es-ES"
espagnol du Mexique	"es-MX"
espagnol d'Argentine	"es-AR"
espagnol de Colombie	"es-CO"
allemand d'Allemagne	"de-DE"
allemand de Suisse	"de-CH"
suédois de Suède	"sv-SE"
portugais du Portugal	"pt-PT"
portugais du Brésil	"pt-BR"

L'instruction sera donc

```
karotz.asr.string("0|1|2|3|4|5|6|7|8|9","fr-FR", function(asrResult));
```

Nous allons adapter notre dernier programme, celui qui bouclait afin de tester différents mots

Voici le main.js que j'explique après

```
include("util.js");
var karotz_ip = "192.168.1.46";//<= ATTENTION : ici votre adresse IP !

var buttonListener = function(event) {
   if (event == "DOUBLE") {
      karotz.tts.stop();
      exit();
   }
   return true;
}

var a_toi_de_jouer = function() {
   karotz.tts.start("donne un chiffre entre 0 et 9", "fr", function(event) {
      if ((event == "CANCELLED") || (event == "TERMINATED")) {
         karotz.asr.string("0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 ", "fr-FR", function(asrResult) {
            var jai_entendu = asrResult.text;
            if (jai_entendu == "<nomatch>")
               setTimeout(6000, function() { a_toi_de_jouer() });

            else
               karotz.tts.start("tu as dis : " + jai_entendu, "fr", a_toi_de_jouer);
         });
      }
   });
}

var onKarotzConnect = function(data) {
   karotz.button.addListener(buttonListener);
   a_toi_de_jouer();
}

karotz.connectAndStart(karotz_ip, 9123, onKarotzConnect, {});
```

On retrouve la fonction « onKarotzConnect » dans laquelle on va autoriser l'évènement « bouton » il va bien falloir sortir du programme quoiqu'on puisse imaginer pouvoir lui dire stop pour qu'il arrête, faites moi penser à essayer plus loin ;)
Cette même fonction enchaîne sur notre fonction principale
L'instruction karotz.asr.string reprend mes explications vues plus haut (grammaire et langue), rien à ajouter, intéressons-nous plutôt au retour de cette instruction.
Lorsque la fonction karotz.asr.string « rend la main » elle a comme résultat asr.text qui est une chaîne contenant ce que nous avons dit ou plus exactement ce qu'il a compris. Il se peut que nous disions rien auquel cas le retour est « <nomatch> », sinon il nous lit ce qu'il a compris.

Sur le principe ça fonctionne plutôt bien si nous parlons distinctement et si nous respectons la règle du jeu à savoir que nous lui donnons un chiffre entre 0 et 9. Que se passe-t-il si on lui dit « zut » (j'ai dit zut, on ne dit pas

de gros mots à son lapin ça n'est pas gentil ☺), « papa », « maman » je vous laisse essayer !

J'ai pensé qu'on pourrait utiliser le retour `asrResult.confident` qui donne une valeur entre 0 et 100, l'explication que l'on trouve sur le site Karotz expliquant qu'au dessus de 30 on peut considérer que le résultat est satisfaisant.

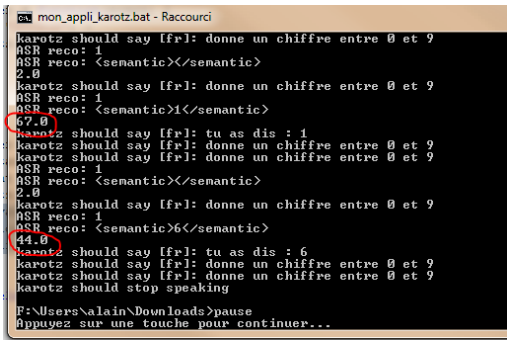
Lorsque je dis papa il comprend des chiffres différents mais avec un asrResult largement supérieur à 30 donc peu probant.

Si vous voulez voir ce asrResult je vous rappelle la fonction de debug « log » qui permet d'afficher une variable

Je l'ai positionnée ici :

```
var jai_entendu = asrResult.text;
log(asrResult.confident);
if (jai_entendu == "<nomatch>")
```

Et voici ce que ça donne



Dans notre cas on ne va donc pas utiliser ce retour mais plutôt ajouter une fonction de confirmation

Je remplace donc :

karotz.tts.start("tu as dis : " + jai_entendu, "fr", a_toi_de_jouer);

Par

karotz.tts.start("As-tu dit : " + jai_entendu + ' ?', "fr", oui_non);

Et j'ajoute la fonction oui_non dans laquelle pour l'instant je fais la même chose selon qu'on dise oui ou non mais on imagine bien que par la suite si je valide mon choix je vais continuer le « traitement » qui consistera à teste si c'est supérieur, inférieur ou égal au chiffre que Karotz aura choisi.

```
var oui_non = function(event) {
  if ((event == "CANCELLED") || (event == "TERMINATED")) {
    karotz.asr.string("oui | non", "fr-FR", function(asrResult) {
      var validation = asrResult.text;
if (validation == "<nomatch>")
        setTimeout(6000, function() { a_toi_de_jouer() });

      else if (validation == "oui")
        karotz.tts.start("je valide ton choix", "fr", a_toi_de_jouer);
      else
        karotz.tts.start("recommence", "fr", a_toi_de_jouer);
    }); }
    }
```

Vous, je ne sais pas, mais moi je suis assez satisfait du résultat.
Le reste du programme ne présente pas de difficultés particulières (enfin pour les pros que vous êtes devenus) ;)
On va d'abord commencer par choisir un chiffre entre 0 et 9, cela n'a plus de secret pour vous car nous avons déjà utilisé cette fonction avec les voix aléatoires. (tutoriel N°2)

```
var mon_nombre_aleatoire = Math.floor(Math.random() *5);
```

Il suffit de remplacer le 5 par 10.

```
var mon_nombre_aleatoire = Math.floor(Math.random() *10);
```

puis dans un premier temps pour tester le programme il est bon de connaitre ce chiffre on ajoute donc en dessous

```
log("le chiffre choisi est : " + mon_nombre_aleatoire) ;
```

Il ne reste plus qu'à compléter le test pour savoir si le chiffre que nous avons dit est supérieur, inférieur ou égal à celui choisi par karotz.
Ça donne ceci

```
var oui_non = function(event) {
    if ((event == "CANCELLED") || (event == "TERMINATED")) {
        karotz.asr.string("oui | non", "fr-FR", function(asrResult) {
            var validation = asrResult.text;
            log(asrResult.confident);
            if (validation == "<nomatch>")
                setTimeout(6000, function() { a_toi_de_jouer() });

            else if (validation == "non")
                karotz.tts.start("recommence", "fr", a_toi_de_jouer);
```

ici la validation est donc oui je regarde donc si le chiffre donné par moi, qui se trouve dans la varaible jai_lu est inférieur à celui choisi par Karotz, si c'est le cas j'annonce que c'est en dessous (trop bas)

```
            else if (jai_entendu < mon_nombre_aleatoire)
                karotz.tts.start("trop bas", "fr", a_toi_de_jouer);
```

ça n'est pas inférieur, alors est-ce supérieur, si c'est le cas j'annonce que c'est au dessus (trop haut)

```
            else if (jai_entendu > mon_nombre_aleatoire)
                karotz.tts.start("trop haut", "fr", a_toi_de_jouer);
```

ça n'est ni supérieur, ni inférieur c'est donc que j'ai trouvé (je rappelle pour ceux qui lirait juste ce passage que les fautes sont volontaires c'est pour que le mot juste soit prononcé, les accents ne sont pas lus)

```
                else
                    log('entendu ' + entendu);
                log('mon nombre ' + mon_nombre_aleatoire);
                karotz.tts.start("Bravo tu as trouvez", "fr", exitFunction);
        });
    }
}
```

Voici donc le main.js

```
include("util.js");
var karotz_ip = "192.168.1.46";
var mon_nombre_aleatoire = Math.floor(Math.random() * 10);
log('chiffre choisi : ' + mon_nombre_aleatoire);

var buttonListener = function(event) {
    if (event == "DOUBLE") {
        karotz.tts.stop();
        exit();
    }    return true;
}
var exitFunction = function(event) {
    if ((event == "CANCELLED") || (event == "TERMINATED")) {
        exit();
    }    return true;
}
var oui_non = function(event) {
    if ((event == "CANCELLED") || (event == "TERMINATED")) {
        karotz.asr.string("oui | non", "fr-FR", function(asrResult) {
            var validation = asrResult.text;
            if (validation == "<nomatch>")
                setTimeout(6000, function() { a_toi_de_jouer() });
            else if (validation == "non")
                karotz.tts.start("recommence", "fr", a_toi_de_jouer);
            else if (jai_entendu < mon_nombre_aleatoire)
                karotz.tts.start("trop bas", "fr", a_toi_de_jouer);
            else if (jai_entendu > mon_nombre_aleatoire)
                karotz.tts.start("trop haut", "fr", a_toi_de_jouer);
                else
                karotz.tts.start("Bravo tu as trouvez", "fr", exitFunction);
  }); }
}
var a_toi_de_jouer = function() {
    karotz.tts.start("cet a toi...", "fr", function(event) {
        if ((event == "CANCELLED") || (event == "TERMINATED")) {
            karotz.asr.string("0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 ", "fr-FR",
function(asrResult) {
                var jai_entendu = asrResult.text;
                entendu = asrResult.semantic;
                log(asrResult.text);
                if (jai_entendu == "<nomatch>")
                    setTimeout(6000, function() { a_toi_de_jouer() });

                else
                    karotz.tts.start("As-tu dit : " + jai_entendu + ' ?', "fr", oui_non);
  });  }  });
}
var onKarotzConnect = function(data) {
karotz.button.addListener(buttonListener);
        karotz.tts.start("donne un chiffre entre 0 et 9", "fr", function(event) {
        if ((event == "CANCELLED") || (event == "TERMINATED")) { a_toi_de_jouer() }
    });
}
karotz.connectAndStart(karotz_ip, 9123, onKarotzConnect, {});
```

Je vous laisse essayer ;)

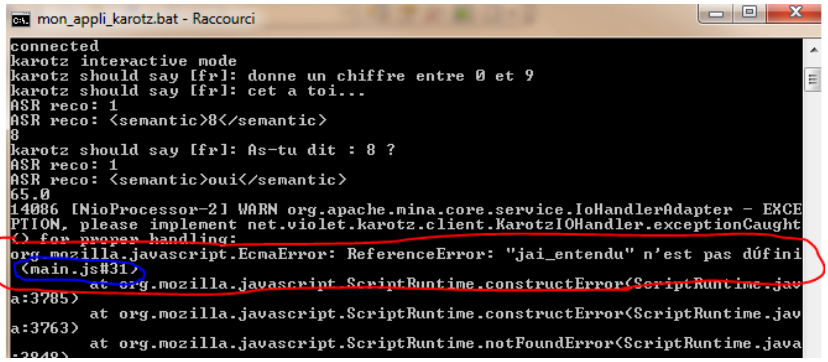
Comment ça je suis sadique? Je sais que ça ne fonctionne pas et pourtant je ne dis rien?

Oui parce que je veux que vous passiez par les mêmes phases que moi, pas par sadisme mais n'est-ce pas la meilleure façon d'apprendre ?

Trébucher, se relever, avancer de nouveau, trébucher…

Je ne sais pas si vous avez fait attention mais dans la fenêtre d'exécution il y a plein de ligne qui s'affichent expliquant qu'il y a au moins une erreur.

Essayons de lire (entre les lignes) pour voir ce qui ne lui plait pas



On note deux choses :

La première c'est qu'il nous dit que « jai_entendu » n'est pas défini (cerclé de rouge)
La deuxième c'est qu'il a trouvé la première fois cette variable dans main.js et que c'est à la ligne 35.
Comptez les lignes avec vos petits doigts et la ligne 35 est la première ligne dans laquelle il y a la variable jai_entendu (mon comptage de ligne ne correspond pas aux vôtres car j'ai supprimé des lignes espaces pour que ça prenne moins de place dans ce tutoriel)

Vous avez dit bizarre ? Comme c'est bizarre !

Etrange cette erreur vous ne trouvez pas ? Pas gentille même. En effet il ne vous aura pas échappé que nous avons déjà utilisé cette variable dans la routine « a_toi_de_jouer » et qu'il n'a rien dit, bien au contraire il nous l'a lu

```
            var jai_entendu = asrResult.text;
        if (jai_entendu == "<nomatch>")
            setTimeout(6000, function() { a_toi_de_jouer() });

        else
            karotz.tts.start("As-tu dit : " + jai_entendu + ' ?', "fr",
oui_non);
```

Alors une idée ?
Malgré toutes vos qualités que je connais bien à présent, vous ne pouvez pas deviner, c'est une particularité de ce langage (qu'on retrouve dans d'autres langages), les variables que nous utilisons doivent tout d'abord être définie, on le fait par « var ma_variable = quelque_chose » jusque là on est d'accord mais on va également différencier une variable « globale » d'une variable « privée »
La variable privée est réservé au module qui l'initialise, ici le module c'est a_toi_de_jouer est donc cette variable n'est connue que par ce module, si je veux l'utiliser dans un autre module comme nous le faisons dans le module oui_non il dira ne pas la connaître.
Alors me direz-vous en vous précipitant, redéfinissons-là dans le module oui_non !
Pas si simple, si on faisait cela elle existerait bien pour le module oui_non mais n'aurait rien à voir avec celle du module a_toi_de_jouer donc le résultat serait tout aussi faussé.
La variable globale, elle, sera initialisée en dehors de tous modules, donc au début du programme, comme notre variable « var karotz_ip » et pourra donc être utilisée dans tous les modules, la valeur de la variable étant celle donnée par le dernier module qui lui a donné une valeur.
Il existe une 3ème voie qui consiste à s'échanger entre modules des variables, lorsqu'on écrit « `var exitFunction = function(event)` » event est une variable passée en paramètre.
Dans le cas qui nous intéresse nous allons définir « jai_entendu » comme variable globale comme ceci :

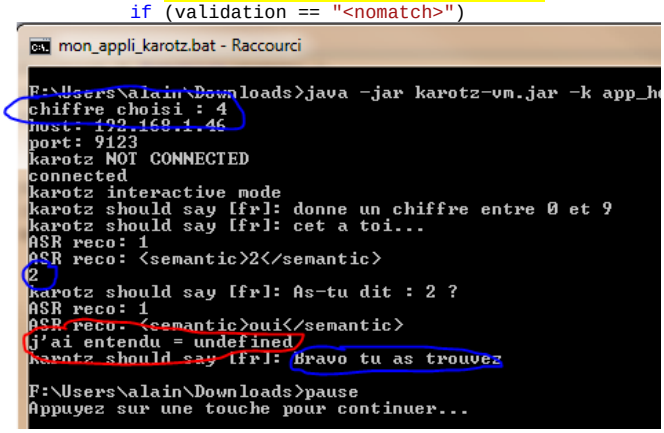
```
include("util.js");
var karotz_ip = "192.168.1.46";
var mon_nombre_aleatoire = Math.floor(Math.random() * 10);
var jai_entendu ;
log('chiffre choisi : ' + mon_nombre_aleatoire);
```

Nous pouvons réessayer.

Magnifique, il n'y a plus d'erreur. Quoique ça ne vous étonne pas vous de trouver à chaque fois du premier coup surtout que si vous regarder le log du chiffre choisi par Karotz il ne correspond pas forcément au notre et pourtant il dit qu'on gagne. Flute, zut et rezut c'était pourtant une bonne chose cette variable globale !

Une idée ? Très bien ta proposition de mettre un log de jai_entendu avant les différents tests, faisons cela de suite :

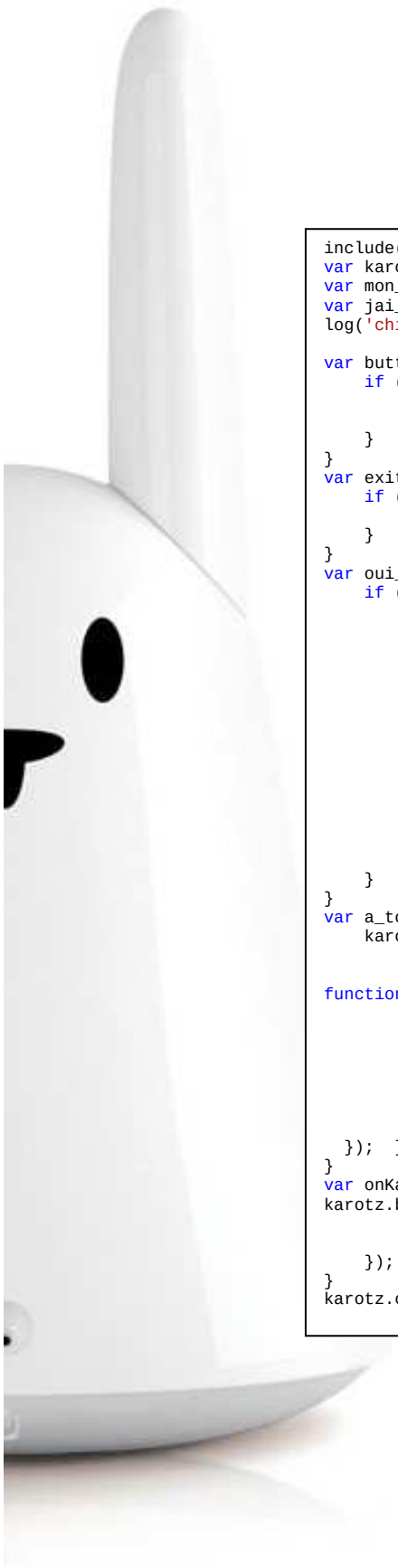
```
var oui_non = function(event) {
    if ((event == "CANCELLED") || (event == "TERMINATED")) {
        karotz.asr.string("oui | non", "fr-FR", function(asrResult) {
            var validation = asrResult.text;
            log("j'ai entendu = " + jai_entendu);
            if (validation == "<nomatch>")
```



Il a choisi 4, j'ai dit 2 et il me dit que j'ai trouvé ah oui mais que dit le log Jai_entendu = undefined

Ça veut dire que la variable existe bien en tant que telle mais qu'elle n'a pas de contenu.

En fait il faut bien comprendre que rien ne nous empêche d'avoir des variables globales et des variables privées et rien ne nous interdit de leur donner le même nom, elles seront bien différentes, c'est juste un peu plus compliqué à s'y retrouver donc on évite ce genre de chose. Et dans notre cas, si je ne m'abuse nous avons toujours `var jai_entendu = asrResult.text;`



dans notre module a_toi_de_jouer.

Il suffit donc d'enlever le mot « var » de cette ligne est tout va rentrer dans l'ordre

Voici le main.js qui fonctionne (toujours perfectible je sais, mais fonctionnel !)

Je vous laisse ajouter le mot « stop » pour qu'il arrête avant la fin, un peu à vous de travailler, non ?

```
include("util.js");
var karotz_ip = "192.168.1.46";//<= ATTENTION ici votre adresse IP
var mon_nombre_aleatoire = Math.floor(Math.random() * 10);
var jai_entendu ;
log('chiffre choisi : ' + mon_nombre_aleatoire);

var buttonListener = function(event) {
    if (event == "DOUBLE") {
        karotz.tts.stop();
        exit();
    }    return true;
}
var exitFunction = function(event) {
    if ((event == "CANCELLED") || (event == "TERMINATED")) {
        exit();
    }    return true;
}
var oui_non = function(event) {
    if ((event == "CANCELLED") || (event == "TERMINATED")) {
        karotz.asr.string("oui | non", "fr-FR", function(asrResult) {
            var validation = asrResult.text;
            log("j'ai entendu = " + jai_entendu);
            if (validation == "<nomatch>")
                setTimeout(6000, function() { a_toi_de_jouer() });
            else if (validation == "non")
                karotz.tts.start("recommence", "fr", a_toi_de_jouer);
            else if (jai_entendu < mon_nombre_aleatoire)
                karotz.tts.start("trop bas", "fr", a_toi_de_jouer);
            else if (jai_entendu > mon_nombre_aleatoire)
                karotz.tts.start("trop haut", "fr", a_toi_de_jouer);
            else
                karotz.tts.start("Bravo tu as trouvez", "fr", exitFunction);
        });
    }
}
var a_toi_de_jouer = function() {
    karotz.tts.start("cet a toi...", "fr", function(event) {
        if ((event == "CANCELLED") || (event == "TERMINATED")) {
            karotz.asr.string("0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 ", "fr-FR",
function(asrResult) {
                jai_entendu = asrResult.text;
                log(asrResult.text);
                if (jai_entendu == "<nomatch>")
                    setTimeout(6000, function() { a_toi_de_jouer() });

                else
                    karotz.tts.start("As-tu dit : " + jai_entendu + ' ?', "fr", oui_non);
  });  }  });
}
var onKarotzConnect = function(data) {
karotz.button.addListener(buttonListener);
        karotz.tts.start("donne un chiffre entre 0 et 9", "fr", function(event) {
        if ((event == "CANCELLED") || (event == "TERMINATED")) { a_toi_de_jouer() }
    });
}
karotz.connectAndStart(karotz_ip, 9123, onKarotzConnect, {});
```

Multimedia

Notre lapin parle, entend, s'allume, bouge les oreilles mais sait-il jouer de la musique ?

Vous souvenez-vous du premier tutoriel ou j'écrivais « roulement de tambour » et bien je n'ai pu m'empêcher de faire ça tout d'abord en pensant à cette publicité pour une pile qui dure qui dure qui dure…

Puis bien entendu je suis allé voir sur notre site favori http://dev.karotz.com/sdk/ à quoi ressemblait l'instruction multimédia. Cette instruction est multiple puisqu'elle peut aussi bien jouer un son en provenance du Web que de la musique y compris une playlist à partir d'une clé USB.

Nous nous intéresserons pour l'instant à l'instruction qui va permettre à notre lapin de jouer du tambour.

Pour cela on utilise notre moteur de recherche favori, on tape roulement de tambour et on regarde un peu ce qui pourrait nous intéresser, personnellement je me suis arrêté sur ce site et je ne regrette pas

http://www.universal-soundbank.com/roulement-tambour3.htm

J'ai retenu ce son là :

http://www.universal-soundbank.com/mp3/sounds/12914.mp3

Et le main.js qui va avec est une simplicité enfantine, à se demander pourquoi ne pas l'avoir mis dans les premiers tutoriels, je vous le livre sans explication.

```
include("util.js");
var karotz_ip = "192.168.1.46";//<= ATTENTION ici votre adresse IP
var buttonListener = function(event) {
    if (event == "DOUBLE") {
        karotz.tts.stop();
        exit();
    }    return true;
}
var onKarotzConnect = function(data) {
    var path = "http://www.universal-soundbank.com/mp3/sounds/12914.mp3";
    karotz.button.addListener(buttonListener);
    karotz.multimedia.play(path, function(event) {
    if (event == "TERMINATED") {
        onKarotzConnect();
        }
    });
}
karotz.connectAndStart(karotz_ip, 9123, onKarotzConnect, {});
```

Voilà une belle entrée en matière je trouve ! Et du coup, ayant gardé une âme d'enfant je me suis amusé à mettre des valeurs au hasard,pour remplacer celle surlignée en vert puis je me suis dit générons cette valeur par programme, prenons un nombre entre 1 et 13000 et laissons faire le hasard, c'est assez cocasse je dois avouer, j'adore le 3042 ceci étant dit ;) et je ne m'en lasse pas, attention malgré tout j'écoute en ce moment une chanson, issue de ce site qui peut surprendre, je ne suis pas responsable du contenu de ce site.
Il faut ajouter la ligne en jaune et modifier la ligne en vert

```
var onKarotzConnect = function(data) {
    var mon_nombre_aleatoire = Math.floor(Math.random() * 13000);
    var path = "http://www.universal-soundbank.com/mp3/sounds/" + mon_nombre_aleatoire + ".mp3";
```

Au fait vous connaissez le pet de lapin ? Non alors écouter ceci http://www.universal-soundbank.com/mp3/sounds/1595.mp3
Heu désolé, il est tard, promis je ne le ferai plus ☺

Le quoi de neuf docteur
http://www.universal-soundbank.com/mp3/sounds/7397.MP3
etc.

Ne pas confondre vitesse et précipitation !

Vous souvenez-vous ? Nous nous sommes quittés avec le tutoriel N°2 sur une promesse d'étudier le chaînage d'instructions.

Nous allons faire quelque chose de simple :

Notre lapin va dire une phrase « j'allume la lumière »

Il va allumer sa lumière en « FADE »

Notre lapin va dire une phrase « je bouge les oreilles »

Il va bouger les oreilles

Notre lapin va dire une phrase « C'est terminé »

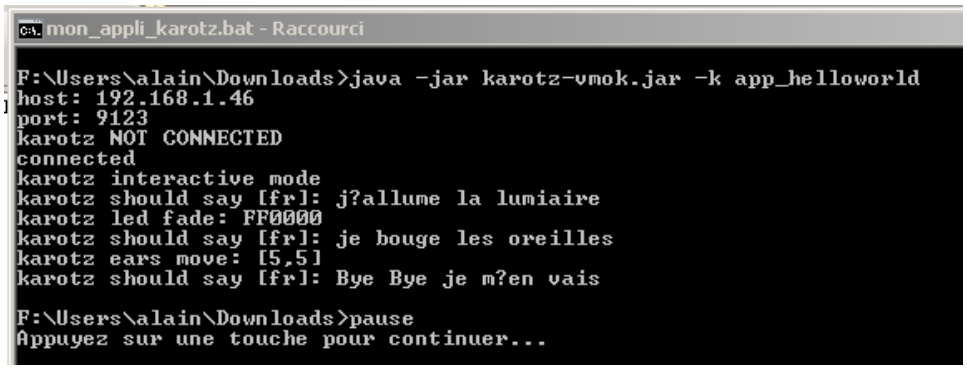
Il quittera le programme.

Tout logiquement on va écrire cela comme ça :

```
include("util.js");
var karotz_ip = "192.168.1.46"//ici votre adresse IP
var buttonListener = function(event) {
if (event == "DOUBLE") {
karotz.tts.stop();
exit();
}
return true;
}
var exitFunction = function(event) {
if((event == "CANCELLED") || (event == "TERMINATED")) {
exit();
}
return true;
}
var onKarotzConnect = function(data) {
karotz.button.addListener(buttonListener);

karotz.tts.start("j'allume la lumiaire", "fr", exitFunction);

karotz.led.fade("FF0000", 5000, exitFunction);

karotz.tts.start("je bouge les oreilles", "fr", exitFunction);

karotz.ears.move(5,5,0)

karotz.tts.start("Bye Bye je m'en vais", "fr", exitFunction);

}
karotz.connectAndStart(karotz_ip, 9123, onKarotzConnect, {});
```

On execute notre fichier batch et et et ….

Pas terrible tout ça ! Et pourtant il a bien exécuté toutes les instructions, la preuve :



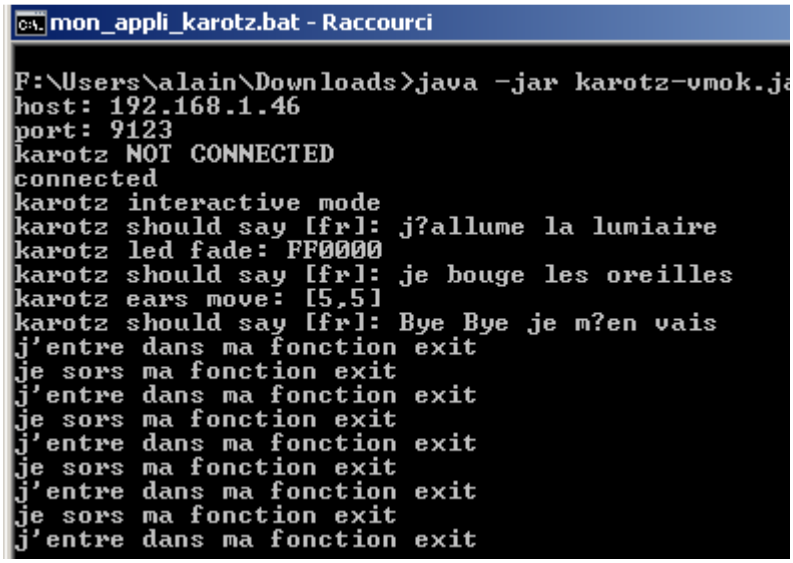
Que se passe-t-il donc ?

Utilisons le log pour voir par où passe notre programme car il ne vous aura pas échappé que chacune de nos instruction contient un appel à exitFunction.

J'insère donc les 2 lignes surlignées en jaune

```
include("util.js");
var karotz_ip = "192.168.1.46"//ici votre adresse IP
var buttonListener = function(event) {
    if (event == "DOUBLE") {
        karotz.tts.stop();
        exit();
    }
    return true;
}
var exitFunction = function(event) {
    log("j'entre dans ma fonction exit");
    if ((event == "CANCELLED") || (event == "TERMINATED")) {
        exit();
    }
    log("je sors ma fonction exit");
    return true;
}
var onKarotzConnect = function(data) {
    karotz.button.addListener(buttonListener);
    karotz.tts.start("j'allume la lumiaire", "fr", exitFunction);
    karotz.led.fade("FF0000", 5000, exitFunction);
    karotz.tts.start("je bouge les oreilles", "fr", exitFunction);
    karotz.ears.move(5, 5, 0)
    karotz.tts.start("Bye Bye je m'en vais", "fr", exitFunction);

}
karotz.connectAndStart(karotz_ip, 9123, onKarotzConnect, {});
```

Et voyons le résultat



On voit très bien que le programme déroule les instructions les unes derrière les autres (en sequentiel donc) et que bien que la première qui dit « j'allume la lumière » ne soit pas terminée il continue sa vie et un moment donné il va exécuter les 4 exitFunction.
Idéalement il faudrait attendre que la première instruction soit terminée avant de démarrer la seconde, attendre que la seconde soit terminée avant de démarrer la 3ème (quoiqu'ici ça n'est pas obligatoire on peut avoir envie que la lumière change pendant la lecture de la phrase) etc

Comme toujours il y a plusieurs façons d'écrire cela, nous allons faire selon nos bonnes habitudes, clarté de lecture et de compréhension.

Donc la fonction appelée dans la première instruction (exitFunction) va désormais s'appeler « attendre_fin_premiere_phrase »
On aura donc :

```
karotz.tts.start("j'allume la lumiaire", "fr", attendre_fin_premiere_phrase);
```

et

```
var attendre_fin_premiere_phrase = function(event) {
    if ((event == "CANCELLED") || (event == "TERMINATED")) {
       //ici je vais mettre l'instruction qui allume la lumière
    }
    return true;
}
```

Comment lire cette fonction ?
Si l'évènement qui m'amène ici (TTS) est annulé (CANCELLED) OU Terminé (TERMINATED)
Alors fait ceci => ici nous allumerons la lumière
Sinon retourne d'où tu viens (en l'occurrence le programme retournera dans l'exécution de l'instruction karotz.tts.start)

Pour l'instruction qui allume la lumière qui possède aussi une fonction event on procèdera de même on aura donc

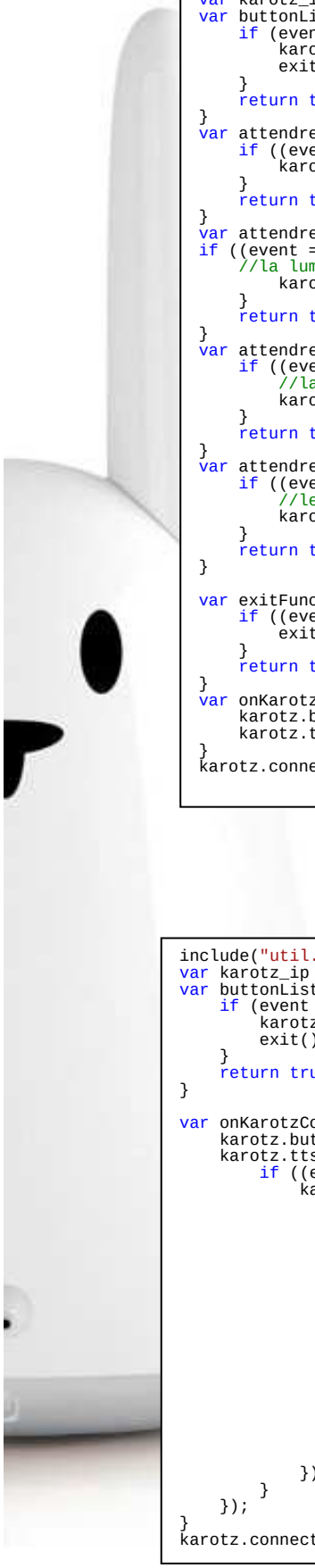
```
var attendre_fin_premiere_phrase = function(event) {
    if ((event == "CANCELLED") || (event == "TERMINATED")) {
        karotz.led.fade("FF0000", 5000, attendre_fin_lumiere);
    }
    return true;
}
```

Et

```
var attendre_fin_lumiere = function(event) {
    if ((event == "CANCELLED") || (event == "TERMINATED")) {
//la lumière a fini de clignoter j'annonce que je vais bouger les oreilles
        karotz.led.fade("FF0000", 5000, attendre_fin_phrase_oreille);
    }
    return true;
}
```

Rebelote avec la fin de cette phrase qui bougera les oreilles (etc)

Voici le code qui fonctionne. J'espère que ce petit exemple vous fera prendre conscience de l'importance de ce que le site appelle les « callbacks », la gestion des évènements de certaines instructions qui vivent leur vie et pour lesquelles il est pratiquement obligatoire de savoir quand elles se terminent et parfois attendre qu'elles se terminent. Beaucoup de bugs viennent d'une mauvaise gestion de cela, on a la chance de le voir dans notre fenêtre dos mais y faisons-nous toujours attention et savons-nous toujours résoudre cela ?



```
include("util.js");
var karotz_ip = "192.168.1.46"//ici votre adresse IP
var buttonListener = function(event) {
    if (event == "DOUBLE") {
        karotz.tts.stop();
        exit();
    }
    return true;
}
var attendre_fin_premiere_phrase = function(event) {
    if ((event == "CANCELLED") || (event == "TERMINATED")) {
        karotz.led.fade("FF0000", 5000, attendre_fin_lumiere);
    }
    return true;
}
var attendre_fin_lumiere = function(event) {
if ((event == "CANCELLED") || (event == "TERMINATED")) {
    //la lumière a fini de clignoter j'annonce que je bouge les oreilles
        karotz.tts.start("je bouge les oreilles", "fr", attendre_fin_phrase_oreille);
    }
    return true;
}
var attendre_fin_phrase_oreille = function(event) {
    if ((event == "CANCELLED") || (event == "TERMINATED")) {
        //la phrase est terminée je peux bouger les oreilles
        karotz.ears.move(5, 5, attendre_fin_oreille);
    }
    return true;
}
var attendre_fin_oreille = function(event) {
    if ((event == "CANCELLED") || (event == "TERMINATED")) {
        //les oreilles ne bougent plus je peux dire la phrase pour partir
        karotz.tts.start("Bye Bye je m'en vais", "fr", exitFunction);
    }
    return true;
}

var exitFunction = function(event) {
    if ((event == "CANCELLED") || (event == "TERMINATED")) {
        exit();
    }
    return true;
}
var onKarotzConnect = function(data) {
    karotz.button.addListener(buttonListener);
    karotz.tts.start("j'allume la lumiaire", "fr", attendre_fin_premiere_phrase);
}
karotz.connectAndStart(karotz_ip, 9123, onKarotzConnect, {});
```

Voici la manière « plus pro » de l'écrire, ça fait exactement la même chose ;) mais la moindre parenthèse manquante, le moindre crochet devient très vite impossible à gérer

```
include("util.js");
var karotz_ip = "192.168.1.46"//ici votre adresse IP
var buttonListener = function(event) {
    if (event == "DOUBLE") {
        karotz.tts.stop();
        exit();
    }
    return true;
}

var onKarotzConnect = function(data) {
    karotz.button.addListener(buttonListener);
    karotz.tts.start("j'allume la lumiaire", "fr", function(event) {
        if ((event == "CANCELLED") || (event == "TERMINATED")) {
            karotz.led.fade("FF0000", 5000, function(event) {
                if ((event == "CANCELLED") || (event == "TERMINATED")) {
                    karotz.tts.start("je bouge les oreilles", "fr", function(event) {
                        if ((event == "CANCELLED") || (event == "TERMINATED")) {
                            karotz.ears.move(5, 5, function(event) {
                                if ((event == "CANCELLED") || (event == "TERMINATED")) {
                                    karotz.tts.start("bye bye je m'en vais", "fr", function(event) {
                                        if ((event == "CANCELLED") || (event == "TERMINATED")) {
                                            exit();
                                        }
                                        return true;
                                    });
                                }
                            });
                        }
                    });
                }
            });
        }
    });
}
karotz.connectAndStart(karotz_ip, 9123, onKarotzConnect, {});
```

Cadeau karotz

En faisant des essais je me suis retrouvé à faire cette petite appli, toute bête notre lapin chante « Meunier tu dors… » et ses oreilles tournent comme les ailes du moulin ;)

```
include("util.js");
var karotz_ip = "192.168.1.46"//ici votre adresse IP

var buttonListener = function(event) {
    if (event == "DOUBLE") {
        karotz.tts.stop();
        exit();
    }
    return true;
}
var exitFunction = function(event) {
    if ((event == "CANCELLED") || (event == "TERMINATED")) {
        exit();
    }
    return true;
}
var onKarotzConnect = function(data) {
    karotz.button.addListener(buttonListener);
    karotz.multimedia.play("http://www.bregeon.net/karotz/meunier.ogg", exitFunction);
    karotz.ears.moveRelative(-50000, 50000, 0)
}
karotz.connectAndStart(karotz_ip, 9123, onKarotzConnect, {});
```

Et voici la fin du 4ème tutoriel. J'espère que vous avez toujours autant de plaisir à les lire et les mettre en pratique.
Dans le prochain nous apprendrons à mettre nos applications en ligne sur le site Karotz.